Predicting residual survival

Nova F. Smedley

2018-11-04

The package `gbmSPM` can be used to do the exact analysis pipeline used in the paper,

Smedley, Nova F., Benjamin M. Ellingson, Timothy F. Cloughesy, and William Hsu. "Longitudinal Patterns in Clinical and Imaging Measurements Predict Residual Survival in Glioblastoma Patients." *Scientific reports* 8, no. 1 (2018): 14429.

See also Supplemental Materials.

Vignette Info

This vignette shows how temporal features and patient covariates are used in predicing residual survival. This depends on the `glmnet` and `caret` R packages, uses dummy patient data, and follows the other vignette: "Generate sequential patterns."

Example

1. **Set the cross-validation and logistic regression parameters**

   - seed - for data partitioning
   - nFolds - number of cross-validation folds
   - cvReps - number of times to repeat cross-validation
   - metric - for selecting hyperparameters
   - lasso - whether to use lasso regularization (lamba)
   - llength - number of lambas to consider
   - lmax - the max. lamba to consider

```
library(gbmSpm)

seed <- 9
nFolds <- 2
cvReps <- 2
metric <- 'rocAUC'
lasso <- TRUE
llength <- 100
lmax <- 0.2
```

2. **Set the parameters used to generate features**

   Previously, we found temporal features via sequential pattern mining (SPM) with `arulesSequences` and placed the patterns in "~/gbm_spm_example". We will also put logit results there as well.

```
# previously created spm patterns parameters
tType <- 'rate'
maxgap <- 60
maxlength <- 2
outdir <- '~/gbm_spm_example'
dataFolder <- 'sup0.4g60l2z2' # previously created spm patterns
dataFile <- file.path(dataFolder, 'featureVectors_rateChange.rds') # data input file
```

```
# output dirs for logit models
prefix <- 'logits'
logitFolder <- file.path(outdir, prefix, dataFolder)
ifelse(!dir.exists(logitFolder), dir.create(logitFolder, recursive = T), F)

# expected ids of train and test, if doesn't exist, will create it
dataPartitions <- file.path(outdir, 'train_test_partitions.rds')

# input path, aka spm data folder
dataPath <- file.path(outdir,'spm',dataFile)
```

Log the experiment:

```
logfn <- file(file.path(logitFolder,
                    paste0(format(Sys.Date(), format="%Y.%m.%d"),
                           '_logit_',getVolTypeName(tType),'.log')),
          open='wt')

sink(logfn, type='output', split=T)

cat(format(Sys.Date(), format="%Y.%m.%d"), '\n')
#> 2018.11.04
cat('pattern dataset:', dataFile, '\n')
#> pattern dataset: sup0.4g60l2z2/featureVectors_rateChange.rds
cat('tumor vol variable type: ', getVolTypeName(tType), '\n')
#> tumor vol variable type:  rateChange
cat('seed: ', seed, '\n')
#> seed:  9
cat('nFolds: ', nFolds, '\n')
#> nFolds:  2
cat('cvReps: ', cvReps, '\n')
#> cvReps:  2

if (lasso) {
  cat('lambda length: ', llength, '\n\n')
  cat('lambda max: ', lmax, '\n\n')
}
#> lambda length:  100
#>
#> lambda max:  0.2
```

3. **Create partitions on cleaned data**

   Data partitioning depends on the fully cleaned dataset and ensures the exact same clinical visits are maintained in training or testing, regardless of any further feature engineering methods.

   Load original data:

```
data("fake_data")
names(fake_data)
#> [1] "person" "demo"   "events"
colnames(fake_data$person)
#> [1] "iois"     "survival" "status"
colnames(fake_data$demo)
#> [1] "iois"      "DOB"       "gender"    "ethnicity" "MSP"       "IDH1"
colnames(fake_data$events)
```

```
#> [1] "iois"         "agent"        "agent_detail" "startdate"
#> [5] "enddate"

survData <- fake_data$person
fake_demo <- fake_data$demo
```

Get the pool of visits for partitioning:

```
# 'c'  tType does not remove initial imaging volumes
# (e.g., rate change needs two visits, the very first one is ignores)
fake_data$events <- cleanData(fake_data$events, tType = 'c')
#>
#>
#>  Cleaning data...
#>
#> ... 33  events were removed due to empty agent_details
#> ... 3  events were removed due to agent_detail = "-"
#> ... 0  events were removed due to mental status = 9
#> ... 0  events were removed due to overallNeuro = 3
#> ... 1097 events were removed due to event being prior to first radiation or no radiation events
#> ... 1 patient(s) were removed
#> ... keep event agents:
#> KPS
#> measurement
#> mentalStatus
#> neuroFunc
#> overallNeuro
#> Radiation
#> Surgery
#> ... 194 events were removed due to no events in agents of interest
#> ... 0 patients were removed due to no events in agents of interest
#> ...these iois were removed:integer(0)

fake_data <- merge(fake_data$events, fake_data$person, by='iois', all.x=T)

# need gender info
fake_data <- prepDemographics(fake_data, fake_demo)

# get survival labels, these also change
fake_data <- prepSurvivalLabels(fake_data)

# clinical ids
fake_data$id <- paste0(fake_data$iois,'.',fake_data$eventID)

# create train and test partitions
# and attempt to maintain gender and survival balance in both parts
part.ids <- getTrainTestPartition(data = fake_data,
                                  database = NULL,
                                  personTable = NULL,
                                  survData = survData,
                                  seed = seed,
                                  verbose = T)
#>
#> ... unique iois in data (passed in):  99
```

```
#> ... gender distribution of  people (with OS < mean OS) in training partition:
#> 1 FEMALE   1 MALE
#>       22       17
#>
#> ... gender distribution of  people (with OS < mean OS) in testing partition:
#> 1 FEMALE   1 MALE
#>        7        5
#>
#> ... gender distribution of people (with OS > mean OS) in training partition:
#> 0 FEMALE   0 MALE
#>       11       26
#>
#> ... gender distribution of  people (with OS > mean OS) in testing partition:
#> 0 FEMALE   0 MALE
#>        3        8
#>
#> ... num people in training partition:  76
#> ... num people in testing partition:  23
#> ... num events in training partition:  1351
#> ... num events in testing partition:  389
#>
#> ... Which patients are in testing and training partitions?:
#>
#> ... Event label distribution in:
#> ......  survivalIn60  training and testing partition, respectively:
#> alive  dead
#>  1312    39
#>
#> alive  dead
#>   375    14
#>
#> ......  survivalIn180  training and testing partition, respectively:
#> alive  dead
#>  1255    96
#>
#> alive  dead
#>   347    42
#>
#> ......  survivalIn270  training and testing partition, respectively:
#> alive  dead
#>  1203   148
#>
#> alive  dead
#>   327    62
#>
#> ......  survivalIn360  training and testing partition, respectively:
#> alive  dead
#>  1169   182
#>
#> alive  dead
#>   311    78
names(part.ids)
#> [1] "train" "test"
```

4. **Read in features for prediction task**

   Get the features generated from SPM:

```
# features and labels for each clinical visit
data <- readRDS(dataPath)
names <- colnames(data)
data <- data.frame(id=row.names(data),data)
colnames(data) <- c('id',names)
cat('...overall samples: ', nrow(data), '\n')
#> ...overall samples:  1500
```

   Convert tumor laterality and tumor location to dummy variables:

```
data <- prepLaterality(data)
data <- prepLocation(data)
```

   Remove clinical visits with no history of length max. gap:

```
data <- removeVisits(data,
                     maxgap = maxgap,
                     maxlength = maxlength,
                     tType = tType,
                     save = F,
                     outDir = logitFolder)
#> ...visits left for training:  1340
```

   Remove unwanted features and labels that should not be in training data for `glmnet`:

```
labels <- getClassLabels()
needToRemove <- c('id','iois','eventID', # remove ids
                  labels, # remove labels
                  'IDH1') # not interested
```

5. **Logistic regression**

   Set labels and formula:

```
ln <- labels[1] # just do one
ln
#> [1] "survivalIn60"

form <- as.formula(paste0(ln,'~.'))
form
#> survivalIn60 ~ .
```

   Set training data and do cross-validation with refitting with the best performing lambda:

```
data$id <- as.character(data$id)
data <- data[data$id %in% part.ids$train,]

aucs <- runCV(data = data,
          formula = form,
          lasso = TRUE,
          llength = llength,
          lmax = lmax,
          labelName = ln,
          needToRemove = needToRemove,
          metric = metric,
```

```
          seed = seed,
          folds = nFolds,
          cvReps = cvReps,
          createModelMatrix = FALSE)
#>
#>
#> ....run  2 CV repeats:
#>
#>
#>
#> .... survivalIn60 CV with seed:  9
#>
#>
#>
#> .... survivalIn60 CV with seed:  18
#> Warning in lognet(x, is.sparse, ix, jx, y, weights, offset, alpha, nobs, :
#> one multinomial or binomial class has fewer than 8 observations; dangerous
#> ground
#>
#>
#> .... survivalIn60 highest averaged results over repeated CV:
#>
#>          prAUC    rocAUC    prAUCSD   rocAUCSD lambda
#> 100 0.01769408 0.5721191 0.01230843 0.04422343    0.2
```